规划在状态上的定义与转移有所不同：一维动态规划状态只有一维；而区间动态规划解决的问题与区间有关，状态有二维，分别表示区间的左端点与右端点；一个状态通常由被它包含且比它更小的区间状态转移而来。区间动态规划通常以区间长度作为动态规划的阶段。阶段(长度)、状态(左右端点)、决策三者按照由外到内的顺序构成三层循环。

例如在一个圆形操场的四周摆放 n 堆石子，现要将石子有次序地合并成一堆，规定每次只能选相邻的 2 堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。求出将 n 堆石子合并成 1 堆的最小得分和最大得分(NOI1995 石子合并)。

该问题可以看作多步决策问题，即决策每一次合并的两堆石子。首先考虑链上的情况，即第一堆和最后一堆无法合并。考虑最后一步，此时对于某个 k，一定是将 $a_1,a_2,\cdots,a_k$ 合在一起形成的石子堆和 $a_{k+1},a_{k+2},\cdots,a_n$ 合在一起形成的石子堆再进行最后一次合并，合成目标石子堆。枚举这个分界点 k，那么合并所有石子的问题在最后一步确定的前提下变成了两个小问题：合并下标在 $1\sim k$ 中的石子和合并下标在 $k+1\sim n$ 中的石子。最优子结构性质是显然的，且由于区间只会越分越小，所以状态也没有后效性，因而可以使用动态规划解决这个问题，算法设计如下。

(1) 状态：$f[i][j]$表示合并第 i 堆到第 j 堆石子的最大得分，$g[i][j]$表示最小得分，$s[i]$表示 $\sum_{j=1}^{i} a_j$。

(2) 状态转移方程：枚举分界点 k，有状态转移方程

$$f[i][j]=\left(\max_{k=i}^{j-1} f[i][k]+f[k+1][j]\right)+\sum_{t=i}^{j} a_t$$

$$g[i][j]=\left(\min_{k=i}^{j-1} f[i][k]+f[k+1][j]\right)+\sum_{t=i}^{j} a_t$$

(3) 初始值：当 $i=j$ 时，$f[i][j]$和 $g[i][j]$均为 0。

(4) 答案：$f[1][n]$，$g[1][n]$。

本算法的时间复杂度为 $O(n^3)$。

代码示例

动态规划求解 NOI1995 石子合并问题的核心代码如下。

```
for (int i = 1; i <= n; i++)
{
    a[i + n] = a[i]; // 将序列复制一份
    f[i][i] = g[i][i] = f[i + n][i + n] = g[i + n][i + n] = 0;
                    // 初始化最底层答案
}
for (int i = 1; i <= n + n; ++i)
{
    s[i] = s[i - 1] + a[i];
}
```

```
for (int len = 2; len <= n; len++)
{
    // 按长度从小到大计算
    for (int i = 1; i <= 2 * n - len; i++)
    {                                  // 枚举区间左端点
        int j = i + len - 1; // 计算右端点
        int sum = a[j] - a[i]; // 计算这一次合并的贡献
        f[i][j] = -inf;
        g[i][j] = inf;
        for (int k = i; k <= j - 1; k++)
        {
            f[i][j] = max(f[i][j],f[i][k] + f[k + 1][j] + sum);
            g[i][j] = min(g[i][j],g[i][k] + g[k + 1][j] + sum);
            // 枚举断点进行转移
        }
    }
}
int mx = -inf;
int mn = inf;
for (int i = 1; i <= n; i++)
{ // 计算所有断环方式下的答案的最大最小值
    mx = max(mx,f[i][i + n - 1]);
    mn = min(mn,g[i][i + n - 1]);
}
printf("%d %d",mx,mn); // 输出答案
```

参考词条

动态规划的基本思路

典型题目

1. NOI1995 石子合并
2. IOI1998 Polygon
3. NOI1999 棋盘分割
4. NOIP2006 提高级 能量项链
5. NOIP2007 提高级 矩阵取数游戏
6. CSP2021-S 括号序列
7. USACO2005Jan Naptime
8. USACO2016Open 248

（谢秋锋）

1.5 数学与其他

1.5.1 数及其运算

1.5.1.1 自然数、整数、有理数、实数及其算术运算

程序中的算术运算与数学中的算术运算具有相同的概念。

程序中的加、减、乘、除的运算优先级与数学中的运算优先级是一样的，先乘除后加减。C/C++中的加号为“+”，减号为“-”，乘号为“*”，除号为“/”。

求余运算也叫取模运算。设 n 和 m 都是整数且 $n\neq 0$，则可以将 m 写为 $m=qn+r$，其中 n、q 和 r 是整数，q 称作商(quotient)，r 称作余数(remainder)，记作 $r=m \bmod n$。在数学中，余数 r 满足 $0\leqslant r<|n|$，在 C/C++中求余符号为"%"，写作 $r=m\%n$，$r=m-qn$，且 $q=\left[\frac{m}{n}\right]$，即 $\frac{m}{n}$ 向零取整的结果。在 C/C++中，如果%左边的数为负数，则模除的结果为负数或者为 0，如果%左边的数为正数，则模除的结果为正数或者为 0，这两种情况下，余数的绝对值均小于除数。

参考词条

整除、因数、倍数、指数、质(素)数、合数

延伸阅读

BRIAN W K，DENNIS M R. C 程序设计语言(第 2 版·新版)[M]. 徐宝文，李志，译. 北京：机械工业出版社，2004：32.

(佟松龄　叶金毅)

1.5.1.2 进制与进制转换：二进制、八进制、十进制、十六进制

进制就是进位计数制，是一种带进位的计数方法，即用有限的数字符号代表所有数值，可使用的符号状态的数目称为基数或底数，基数为 b，则称为 b 进制。

具有 n 个数位的 b 进制数可以表示为 $(d_n d_{n-1}\cdots d_2 d_1)_b$，其中 d_i 表示数字中第 i 位的数值，该 b 进制数的数值为

$$d_n\times b^{n-1}+d_{n-1}\times b^{n-2}+\cdots+d_2\times b+d_1$$

在数制中，各位数字大小与所在位置有关，固定位置所对应的单位值，称为位权。

二进制是以 2 为基数的计数系统，只有 0、1 两种基本符号状态来表示数值，当表示更大数值时向高位进位，即逢二进一。

八进制、十进制、十六进制的基本原理与二进制类似，只是各进制下基本符号状态数不一样，特别是十六进制，其使用的是0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F十六种状态，十六进制下的10到15是以A到F对应表示。

书写时为了区分不同进制，通常将基数注在右下方，二进制通常在右下方注上基数2，或在后面加B表示，例如$(101011)_2$、101011B；八进制用下标8或在数据后面加O表示；十六进制用下标16或在数据后面加H；十进制使用时可以不加标注，或加后缀D。

十进制数转二进制，可将十进制数划分为整数部分和小数部分(若没有小数则省略此步)。整数部分用除二取余法，小数部分用乘二取整法。

- 除二取余法：用2连续除十进制整数，直到商为0，逆序排列余数即可得到该十进制数整数的二进制表示。
- 乘二取整法：用2乘十进制小数，可以得到积，将积的整数部分取出，再用2乘余下的小数部分，如此循环，将每次取出的整数部分顺序排列，得到小数的二进制表示。

有的小数部分乘二取整可以无限运算下去，所以根据精度只需截取足够位数即可停止运算。

十进制整数转八进制即用8连续除十进制数，直到商为0，逆序排列余数即可得到十进制数的八进制表示；十进制整数转十六进制即用16连续除十进制数，方法与以上转换类似。

二进制转十进制可使用按位权展开法，即将二进制数按位权展开，各个二进制位乘以对应位置的位权，并相加求和，即得到对应的十进制数。

例如二进制数10110.1101B，将其各位上的系数乘以位权并求和，即得到对应十进制数：

$$1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}=22.81625$$

以此类推，任意b进制数按照位权展开、相加，即可得到十进制数。二进制数的位权是以2为底的幂，八进制数的位权是以8为底的幂，十六进制数的位权是以16为底的幂。数位由高到低，以降幂的方式排列。

代码示例

整数n转为b进制数的核心代码如下。

```
int x = 0;
while (n != 0)
{
    a[x] = n %b;
    x++;
    n = n / b;
}
```

参考词条

1. 自然数、整数、有理数、实数及其算术运算
2. 位运算：与(&)、或(|)、非(~)、异或(^)、左移(<<)、右移(>>)

延伸阅读

NELL D，JOHN L. 计算机科学概论(原书第5版)[M]. 吕云翔，刘艺博，译. 北京：机械工业出版社，2016：22-31，42-44.

(佟松龄　叶金毅)

1.5.2　初等数学

1.5.2.1　代数(初中部分)(略)

1.5.2.2　几何(初中部分)(略)

1.5.3　初等数论

1.5.3.1　整除、因数、倍数、指数、质(素)数、合数

如果整数 b 除以非零整数 a 的商为整数，且余数为零，则称为 b 能够被 a 整除，其中 b 为被除数，a 为除数，记作 $a \mid b$，读作“a 整除 b”或者“b 能够被 a 整除”。否则称为 b 不能被 a 整除，记作 $a \nmid b$。

如果整数 b 能够被 a 整除，则称 b 是 a 的倍数，a 是 b 的因数，因数也称为约数。

(1) 只有在整除的条件下，才有因数和倍数的概念。

(2) 因数和倍数是相互依存的，不能单独存在。只能说某个数是另一个数的因数或者倍数，而不能说某个数是因数或某个数是倍数。

(3) 两个整数存在倍数和因数的关系是相互的，如果 a 是 b 的因数，则 b 一定是 a 的倍数。

形如 $a^n=b$ 的表达式为指数式，其中 a 为底数，n 为指数，b 为幂(指数运算的结果)。

当指数 $n=0$ 时，$a^0=1(a\neq 0)$。当指数 $n=2$ 时，称为平方。当指数 $n=3$ 时，称为立方。当指数 n 为正整数时，a^n 表示 n 个 a 连乘，也即 $a^n=\underbrace{aa\cdots a}_{n个}(n\in N^*)$。

指数运算性质包括：

(1) $a^n a^m=a^{n+m}$；

(2) $(a^n)^m=a^{nm}$；

(3) $(ab)^n=a^n b^n$；

(4) $\left(\frac{a}{b}\right)^n=\frac{a^n}{b^n}$；